An Evaluation of the Intel 2920 Digital Signal Processing Integrated Circuit

J. Heller
Communications Systems Research Section

The Intel 2920 integrated circuit is a device for digital signal processing. Consisting of a digital-to-analog converter, accumulator, read-write memory and UV-erasable read-only memory, the circuit can convert an analog signal to a digital representation, perform mathematical operations on the digital signal and subsequently convert the digital signal to an analog output. In addition to the 2920 circuit, Intel also offers development software tailored for programming the 2920. This is a report of experiences with the 2920 circuit and its software.

I. Introduction

Intel's 2920 digital signal processor is the first commercially available integrated circuit which allows the user to implement custom digital signal processing algorithms. (Two other circuits, American Microsystem's S2811 and NEC's UPD7720 are not available yet.) Unlike the 7720, Intel's device performs both analog-to-digital and digital-to-analog conversion on the chip. All the circuits are designed for the high-speed multiplications and additions which are the basis of digital filtering algorithms.

The 2920 is being evaluated for use as a low-pass filter. This report describes the experiences obtained from the circuit itself and the support software from Intel. Although the software proved to be exceptional, the chip has drawbacks of a serious nature which could limit its use.

II. 2920 Functional Description

The functional block diagram is reproduced in Fig. 1. The chip can accept up to 4 analog signals at ±1 to ±2 volts peak depending on an external reference voltage. These signals can be multiplexed to a sample-and-hold circuit which, in conjunction with a digital-to-analog converter, can be configured, under programmed control, to act as an analog-to-digital converter with 8-bit precision plus a sign bit. The digital section of the 2920 features a 40-word, 25-bit wide read/write memory (RAM), and a shift register capable of shifting up to 13 bits to the right or 2 bits to the left in a single instruction cycle. The shifter is connected to one port of an arithmetic and logic unit (ALU), while the other ALU input port receives data from the RAM. The output of the ALU returns to the RAM. The ALU features addition, subtraction, absolute value, "exclusive or," "and," "limit" and other operations. Individual bits can also

be tested, although not in the ALU. The digital-to-analog converter used for the conversion of analog signals is also used to convert the digital representation back to the analog form for one of eight sample and hold circuits. The chip sections are controlled by a program stored in an UV erasable read-only memory (EPROM) which can accommodate 192 24-bit instructions. All instructions take the same amount of time to execute which, for the 2920-16, is 600 nsec.

III. Programming Considerations

The 2920 program runs as an endless loop with no jumps allowed. One instruction (EOP) can be used at the end of the program to jump to the beginning if not all 192 EPROM locations are needed. This instruction must be located at word addresses divisible by 4 and must be followed by 3 NOPs.

In order to convert analog signals to digital values, an instruction sequence consisting of a group of IN codes followed by a CVT-NOP-NOP group for each bit is required. Thus, a 9-bit conversion requires 35 instructions. To output an analog signal, approximately 7 NOPs are needed prior to the OUT instructions to allow the digital-to-analog buffer amplifier to settle. Seven OUT instructions are recommended in order to ensure that the sample-and-hold circuit captures the correct value.

Instructions can be executed on a conditional basis depending on the state of a specified bit in the DAR register. If the instruction condition is not fulfilled, a NOP is executed instead. During program segments in which the DAR is not used, it is possible to execute IN or OUT instructions simultaneously with ALU instructions.

IV. 2920 Inadequacies

The 2920 circuit has a number of problems which can cause difficulties of varying degrees. These include faulty instructions as well as gain and offset errors. It should be noted that the following comments apply to the 2920-16 (600-nsec instruction cycle time). The 400-nsec part (2920-10) is unavailable.

Two instructions fail to work properly. The ABA instruction, which takes the absolute value of the contents of a source register and then adds it to the contents of a destination register, does not execute correctly at 600 nsec. In order to use this instruction, the clock rate must be reduced to 4 MHz or less; otherwise the instruction should be replaced by the sequences ABS and ADD. The second faulty instruction, EOP, is used as a jump from the end to the beginning of the program, a useful feature when not all 192 ROM locations are needed. The EOP instruction requires a pull-up resistor on pin 21 where the pin is connected internally to an open drain MOS transistor. The pin is also an input, presumably to a MOS transistor gate, for the reset function. Thus in order for the EOP instruction to operate properly, the voltage at pin 21 must go low at the proper time in order to reset the program counter. At present, in addition to the pull-up resistor at pin 21, a 15-pf capacitor must be connected between pin 21 and ground for the EOP instruction to work. According to Intel, another open drain output at pin 22, OF, does not operate properly. This pin goes low when the accumulator overflows.

The 2920 is subject to analog complaints as well. Given a reference voltage of +1.000 volts, the maximum analog output voltage is approximately ±0.93 volts instead of ±1 volt. For a 2-V reference, the range is ±1.8 V. The 2920 also suffers from a large output offset error which varies with the signal sign and magnitude. As an example, the offset error measured for one 2920 output channel ranged from +13 mV at -0.9 V to -62 mV at 0.0 V to -77 mV at +0.9 V. The results of an experiment point to a nonlinearity in the sample-and-hold buffers driving the output pins. Whatever the cause, errors of this size are serious in low-pass filter applications; hence a number of fixes have been suggested.

With the first method an uncommitted output pin is connected to an unused input. The desired output signal from the DAR is first sent to the previously uncommitted output channel and then read-in to the DAR via the previously unused input channel. The difference between the new value in the DAR and the original DAR contents is used as a correction – the corrected value is then output to the appropriate channel. The success of this method depends on how closely the "uncommitted" output channel matches the channel which outputs the corrected signal.

Another scheme chops the signal by alternately multiplying it by +1 then -1. The output is capacitively coupled to a rectifier which detects the waveform envelope, followed by a low-pass filter for smoothing out the sampling pulses. The chopping can be accomplished by first loading the DAR with the true digital signal representation and converting it to its analog value. Then the same number is exclusive or'd with -1, loaded into the DAR and converted to its analog value. Both signals are alternately output on the same channel.

A third method is based on "correction curve" code appended to the filter program. The curve is determined experimentally by programming the 2920 to act as a wire and then applying accurately measured input voltages over the range of operation and measuring the output voltages. These data can be used with the Intel-supplied signal processing

compiler software to generate code which multiplies the digital value by a correction factor prior to loading into the DAR for output. This approach requires a substantial amount of read-only memory for good accuracy.

A revision of the 2920 from the present "D step" to an "E step" version is planned by Intel for this summer. To what extent the above problems will be solved remains to be seen.

V. Program Development: Hardware

The only necessary hardware for 2920 programming is an EPROM programmer (and an ultraviolet light EPROM eraser). However, in order to take advantage of the support software, an Intel Intellec Development System is required. The 2920 applications software is shipped on both single and double density floppy disc formats. Software modules are included which utilize the floating-point hardware board option in the Intellec; otherwise, software is used for these calculations. A special PROM programming card and adapter socket are also required for the Intellec PROM programmer.

VI. Program Development: Software

Three programs are used to develop code for the 2920: the 2920 signal processing applications software/compiler, a 2920 assembler and a simulator. The applications software/compiler program is the centerpiece of the 2920 software development scheme. The use of the term "compiler" is appropriate as the program enables the engineer to examine and specify filters with familiar design terminology which the compiler in turn converts into 2920 assembly code.

The compiler provides the user with three planes, s, Ts, and z, and the commands for creating, moving and deleting poles and zeros on these planes. The s plane is used to characterize any circuitry preceding the 2920 in the signal path. This is a particularly useful feature for modeling anti-aliasing filters. Poles and zeros are also placed in the Ts and z planes, and it is these values which are used to represent the 2920 digital filter. The Ts plane is provided for designers who prefer a sampling plane that corresponds more closely to the s plane than the z plane does. Prior to code generation, the poles and zeros placed on the Ts plane are mapped to the z plane using the transform $e^{2\pi\ Ts(x+jy)}$ where T is the sampling period and $s(x+jy)$ are the pole/zero coordinates on the Ts plane. Readers may recognize this mapping as the "matched z transform." Finally, since the 2920 uses sample-and-hold circuits on the analog outputs, it is possible to model this effect as well by using the HOLD ON command.

The usefulness of the compiler is apparent after the poles and zeros for the signal path have been specified, for then it is possible to graph, with simple commands, gain and phase as functions of frequency. In addition, the step and impulse responses of the filter can be displayed. After a suitable pole/zero constellation is determined, the compiler is used to generate the 2920 assembly code. It is also possible to generate code for implementing operations such as limiting or rectification.

Another powerful capability of the compiler is the macro feature. A macro is a compiler directive which executes sequences of compiler commands upon invocation of the macro name. Macros are utilized where repetitive sections of code are required or for designing a family of filters, where the structure is the same but parameters such as cutoff frequency and ripple vary. The compiler comes with a number of macros including ones for generating Butterworth and Chebyshev filters. The user can also write and save new ones.

Another helpful aid is an arithmetic interpreter which allows the user to enter a Fortran-like mathematical expression for evaluation. An extremely valuable option is the ability of the compiler to create a file which contains a record of all subsequent compiler commands and responses. This is an aid for reviewing previous design sessions.

The compiler has some drawbacks which the user should be aware of. First, the matched z transform which is used to map from the Ts to z plane can give incorrect results (e.g., when the zeros in the Ts plane have center frequencies greater than half the sampling frequency (Ref. 1)). Another difficulty occurs when the compiler generates 2920 assembly code. The code for each pole and/or zero is produced independently of the previous poles and zeros; hence it is necessary to use equivalence statements such as OUTP1 EQU INPZ2 to link the output variable of a pole or zero to the input variable for the next pole or zero. A similar procedure is required for connecting the analog input and output routines to the rest of the filter code. Finally, the gain calculation occasionally suffered from overflow.

The 2920 assembler is an easy-to-use program that can be invoked, along with options, with a single command. The assembler terminates after the creation of an object file suitable for loading into the 2920 EPROM and a list file showing the original assembly code and its address assignment in the 2920. An important debug option is available which, when used in conjunction with the simulator (see below) enables the designer to reference 2920 memory locations with the same symbolic names as specified in the assembly code provided the names are preceded by "RAM." or "ROM." ROM instructions can be altered using symbolic code also.

The 2920 simulator is the other significant piece of software that Intel provides. Using as input the object code generated by the assembler, the simulator interprets the 2920 code instruction by instruction. It is similar in operation to software debuggers as it provides a breakpoint, a breakpoint qualifier (i.e., those conditions which must be met at, and prior to, the breakpoint), and a trace feature which saves the values of selected analog inputs and outputs as well as the contents of 2920 memory locations as a function of time or iteration number. Also the simulation can be halted so that chosen memory locations, etc., may be displayed. The simulator also provides a "calculator" feature similar to the compiler; however, it does not accept numbers in scientific format (e.g., 4.2E5), unlike the compiler version, an omission which is quite annoying.

It is recommended that the simulator be used in conjunction with a floating point hardware board in order to speed up calculations. Another time-wasting operation is the simulation of input and output conversion codes. Thus, in order to reduce execution time it is worth the bother to create a "streamlined" version of the assembly code for submittal to the simulator. Even this will not be satisfactory for simulations which must evaluate the 2920 over minutes, not just seconds. One simulation carried out during the course of the project required 4 days to generate 15 minutes worth of data. Users should also be aware that the RAM locations in the simulator representing the 2920 RAM are not cleared prior to execution of the simulation. It appears, however, that the 2920 circuit RAM is not cleared and indeed has random values on power-up, so this is perhaps a "realistic" (if unwitting) simulation. The simulator allows users to initialize RAM locations if desired.

The simulator's analog output values include an added offset error representing the error incurred by conversion of the two's complement digital representation to a sign magnitude format in the 2920 D-to-A converter. As the offset errors seen on actual 2920 devices are large, the simulator offset error is negligible.

One other inconvenience is that the simulator must be informed of the time required to execute one complete pass of the code. It would be desirable to specify the execution time for a single instruction instead and let the simulator figure out the rest. Nevertheless, the simulator along with the compiler and assembler are indispensible for the development of 2920 code in a reasonable amount of time.

All the software operated flawlessly. None of the programs entered states from which there was no return. Some of the terminology could have been coordinated better. One LOADs a file in the simulator but INCLUDEs a file in the compiler, for example, and the "calculator" feature was not identical in both the simulator and compiler. Nevertheless, these are small gripes about smooth-operating software. The compiler and simulator have HELP commands which provide the user with explanations about program commands. These were not found to be useful.

VII. Software Documentation

Three manuals accompanied the software – one each for the assembler, compiler and simulator. There is very little overlap of information.

The 2920 Assembly Language Manual (Intel No. 9800987-01) discusses the 2920 functional elements, instructions and assembly language symbols and format. One chapter describes design techniques including multiplication and division methods. The bit patterns of the 2920 instructions are presented in an appendix. Two other appendices are devoted to carry and overflow and two's complement data handling considerations. This information is important and is not repeated elsewhere.

The 2920 Signal Processing Applications Software/Compiler User's Guide (No. 121529-003, Rev. B) is the least accessible of the three manuals. The guide does not give a satisfying overview of the design approach – instead, a "sample session" of 19 pages is presented for the reader to glean whatever information he can. Most of the manual is devoted to compiler command definitions, which are not always clear. Listings are included for the macros which have been supplied with the compiler. Some of the more useful data are in the appendices under "Notes and Cautions," "Design of Complex Digital Filters Used in the 2920," and "Formulas Used by the SPAS20 Compiler." In short, it is more of a reference manual than a user's guide.

On the other hand, the "2920 Simulator User's Guide" (No. 9800988-02 Rev. B) is easier to assimilate due, in part, to its brevity. The commands are not involved so the user can get started quickly.

A fourth document is necessary for programming the 2920: "Universal PROM Programmer User's Manual" (No. 9800819-01). Sections 5-33 through 5-38 are devoted to programming the 2920 EPROM. Although somewhat understandable, an example showing the required commands would be of great value if only as a time-saver. To summarize, users will find it necessary to familiarize themselves with all three software manuals.

VIII. Other Documentation

Intel publishes two other pieces of literature which are worth reading. The "2920 Analog Signal Processor Design Handbook" is a nonmathematical tutorial in digital signal processing followed by a discussion of the 2920 architecture. Subsequent chapters deal with realizing arithmetic functions (similar to the discussion in the assembly language manual), oscillators and nonlinear functions, different filter types, implementation techniques, etc. Design examples are shown including one section on breadboarding. A weak chapter on the software support programs (i.e., compiler, simulator, etc.) is included. For designers who are considering the 2920 as a filter candidate and need a detailed discussion of the 2920, this book is the recommended one.

The final document which should be consulted is the "2920-16 Signal Processor" data sheet (October 1980, marked "Preliminary"). Unlike the earlier data sheet "2920-10" (which incidentally refers to a 10-MHz 2920 not mentioned on the newer sheet) this paper provides specifications on the analog performance of the device, although no mention is made of the inadequacies cited in Section IV above. The document presents the only good discussion of input and output code-timing requirements.

IX. Conclusion

The concepts embodied by the 2920 device represent a formidable tool for realizing compact digital filters. The software support is excellent. However, new users should be cognizant of 2920 device problems.

Reference

1. Rabiner, L., and Gold, B., *Theory and Application of Digital Signal Processing*, New York, 1975, pp. 224-6.

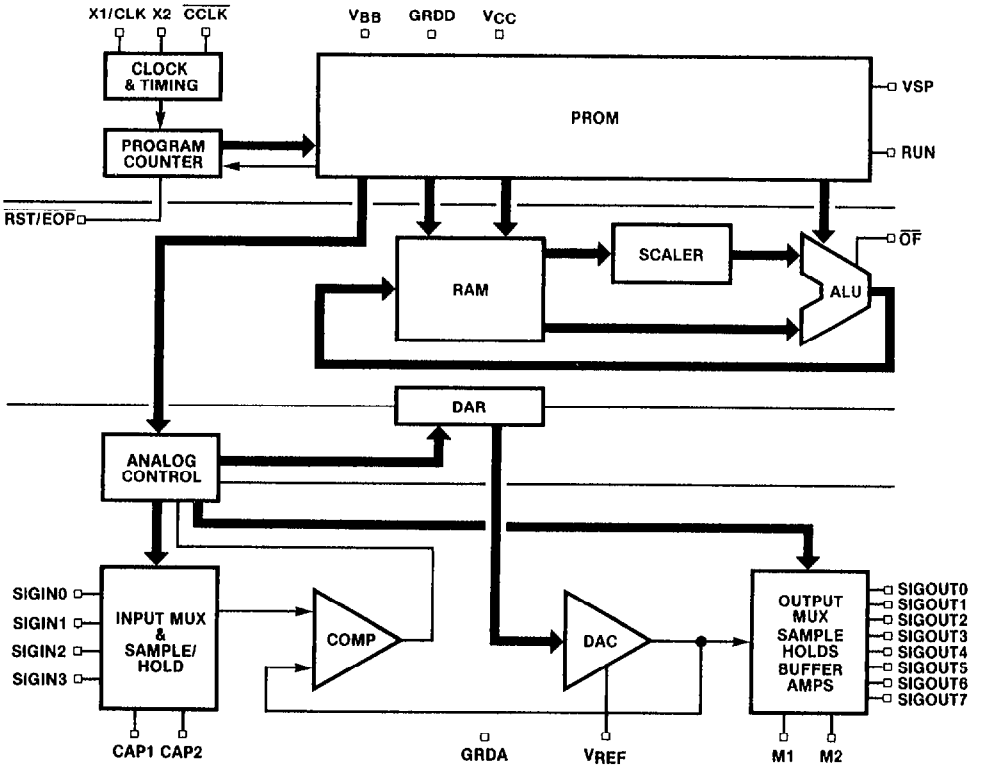

Fig. 1. Block diagram of 2920 signal processor